UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/821,586	04/09/2004	Albert B. Barabas	11811-008002	1971

26161 7590 08/20/2008

FISH & RICHARDSON PC
P.O. BOX 1022
MINNEAPOLIS, MN 55440-1022

EXAMINER	
TO, BAOQUOC N	

ART UNIT	PAPER NUMBER
2162	

MAIL DATE	DELIVERY MODE
08/20/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/821,586	BARABAS ET AL.	
	Examiner	Art Unit	
	BAOQUOC N. TO	2162	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 June 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-60 and 68-83* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-60 and 68-83* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *06/10/2008*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1, 23, 52 and 57-61 are amended in the amendment filed on 06/10/2008. Claims 1-61 and 68-83 are pending in this application.

Information Disclosure Statement

2. The information disclosure statement (IDS) submitted on 06/10/2008. The submission is in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement is being considered by the examiner.

Specification

3. The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter. See 37 CFR 1.75(d)(1) and MPEP § 608.01(o). Correction of the following is required: physical article or object.

Response to Arguments

4. Applicant's arguments with respect to claims 1, 52, 57 and 68 have been considered but are moot in view of the new ground(s) of rejection.

Applicant argues "Hapner does not describe and would not have made obvious the feature of claim 57. For example, Hapner does not disclose a job that includes a pointer to data in the region of database..."

The examiner respectfully disagrees with the above argument. As explained in the Office Action, Hapner discloses a log file which indicate the pointer for writing the data into a part of database (col. 9, lines 55-60).

Applicant also argues "therefore, each record contains filenames associated with a transaction. However, different transactions can perform action on the same file sequentially, and thus different records or templates associated with different transaction can include the same filenames. Accordingly, the "index" of Zaiken does not identify contention spaces that are distinguish from other contention spaces of jobs that dot not have competing requirements to write into the regions of the database. "

The examiner respectfully disagrees with the above argument. As known in the art each job process includes a job identification number. Index in Zaiken as disclosed would distinguish plurality of jobs and determine which jobs require to write into the database using the locking mechanism. Therefore, such uses of index which when combine would yield expectable result to allow high priority job to be executed as suggested by KRS (Moreover, in view of the guidance provided by the Supreme Court in *KSR* decision, the a patent claim is prima facie obvious if "some motivation or suggestion to combine the prior art teachings" can be found in the prior art, the nature of the problem, or the knowledge of a person having ordinary skill in the art. See the recent Board decision *EX parte Smith*, --USPQ2d--, slip op. at 20, (Bd. Pat. App. & Interf. June 25, 2007 (citing *KSR*, 82 USPQ2d at 1396) (available at <http://www.uspto.gov/web/offices/dcom/bpai/prec/fd071925.pdf>).

Applicants also argue "Hapner does not describe or would have made obvious the features of claim 68. For example, Hapner does not accept task from the task source for concurrently execution by multiple processors..."

The examiner respectfully disagrees with the above argument. As explained in the Office Action, Hapner discloses a mutex is created to corresponding to a piece of code, a portion data, some state, etc... when a thread has locked a mutex, it is said to "own" the locked mutex. In order for other threads to own the mutex, the first thread (i.e. the thread that locked the mutex) must unlock it. Thus mutexes provide a mechanism by which the programmer can control the serialization of multiple threads, ensuring that steps occur in a desired order and that the state corresponding to the mutexes are maintained in a consistent manner" (col. 10, lines 11-22). This suggests mutexes is mutual exclusion lock to allow the transaction holding the mutex lock to execute first and once the transaction is committed the data is not loss. In Hapner, the system includes multiple threads to write into the database using plurality of processors with in the same database (col. 9, lines 54-65). The mutexes are solution for tasks confliction, a certain threads which holds the mutex lock having priority over the other threads without holding mutex (col. 10, lines 11-22).

Applicant argues "Fleischman does not remedy Hapner's deficiencies. As explained above, Fleischman's multiple threads inquire of different portions, for example, the keys, the first and second portions, asynchronously."

The examiner respectfully disagrees with the above argument. In Fleischman, the all multiple threads are concurrently executed in the same database (col. 5, lines 33-

36). This suggests some level of guarantee that the all threads will be executed concurrently. Although, Fleischman was explicitly regarding to execute multiple threads concurrently, but Fleischman was not negated or exclude from executing from plurality of threads within the same region of database. Therefore, Fleischman also implicitly discloses thread with higher priority which will guarantee the execution without the loss of data.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-27, 32-43 and 48-51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hapner et al. (US. Patent No. 5,727,203) in view of Cheriton (US. Patent No. 5,666,514) and further in view of Somani et al. (US. Patent No. 5,524,212) and further in view of Fleischman (US. Patent No. 6,507,847

Regarding on claim 1, Hapner teaches a method comprising:

maintaining a database that stores data persistently (col. 7, lines 15-16),

accepting tasks from the task sources, at least some of the tasks having competing requirements (multiple threads competing) for user of regions of the database (resources).

Hapner does not explicitly teach data included in a given region not being available for simultaneous access for writing by more than one of the tasks having competing requirements and data included in different regions being available for simultaneous access for writing by more than one of the tasks having competing requirements, assigning each of the region to a available processor, each of the regions being assignable to any of the processors, defining, for each of the tasks, jobs each of which requires write access to regions that are to be accessed by no more than one of the processors and distributing the jobs for concurrent execution by the associated processors. Cheriton discloses data included in a given region not being available for simultaneous access for writing by more than one of the tasks having competing requirements and data included in different regions being available for simultaneous access for writing by more than one of the tasks having competing requirements (...a read operation or an additional shared ownership mode which allows multiple processors and caches to read the data concurrently...) (col. 6, lines 21-59). This suggests some part of the data cannot be access based on an exclusive lock and other data can be accessed concurrently by plurality of processors. Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify teaching Hapner to include some part of the data cannot be access based on a

exclusive lock and other data can be accessed concurrently by plurality of processors as disclosed by Cheriton in order to allow the multiple threads to access data. Hapner does not explicitly discloses assigning each of the region to a available processor, each of the regions being assignable to any of the processors, defining, for each of the tasks, jobs each of which requires write access to regions that are to be accessed by no more than one of the processors and distributing the jobs for concurrent execution by the associated processors. On the other hand, Somani teaches assigning each of the region to a available processor, each of the regions being assignable to any of the processors, defining, for each of the tasks, jobs each of which requires write access to regions that are to be accessed by no more than one of the processors "as a user writes an INSIGHT program 272 defining application tasks and the number of processor used for each task...Each configuration specification file defines a set of generic processors and the jobs partitioned among them" (col. 27, lines 56-62). This teaches associated task with each processor and only one task for that processor. Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify Hapner and Cheriton to include the one task for one processor as taught by Somani to allow one processor to read or write the data to the database to keep the data consistency. Furthermore, Fleischman also discloses a large number of threads can be executing in simultaneously within the database. The term "simultaneously" is used in the context known to programmers similar to "multithreading" i.e. multiple threads do not execute in perfectly simultaneous manner unless the server has parallel processor (col. 5, lines 8-20) and multiple small "read" and "write" commands to a disk

drive...(col. 5, lines 33-36). Fleischman disclose the execution of the multiple write threads, and each of the processor is assign to a specify write to a specific portion of the database. Furthermore, in the write thread no other processor to be able to access to the lock region which is accessed by other processor. Therefore, Fleischman implicitly discloses the concept of assigning the assigning each of the region to a available processor, each of the regions being assignable to any of the processors, defining, for each of the tasks, jobs each of which requires write access to regions that are to be accessed by no more than one of the processors and distributing the jobs for concurrent execution by the associated processors. Therefore, it would have been obvious to one ordinary skill in art at the time of the invention was made to modify both Hapner, Cheriton and Somani system to include executing of the multiple write threads, and each of the processor is assigned to a specify write to a specific portion of the database and no other processor be able to access to the lock region which is accessed by other processor in the write thread as disclosed by Fleischman to resolve any conflict of accessing the database.

Regarding on claims 2 and 53, Hapner teaches the stored data includes items of the database comprises objects in an object database (col. 9, lines 1-3).

Regarding on claims 3 and 54, Hapner teaches the stored data includes data items that are provided as objects to an object-oriented application (col. 7, lines 25-30).

Regarding on claims 4 and 55, Hapner teaches an objected relational broker provides persistent storage of objects for an object-oriented application (col. 7, lines 34-35).

Regarding on claims 5 and 56, Hapner teaches the data is stored in a relational database with object-oriented extensions (col. 7, lines 1-10).

Regarding on claim 6, Hapner teaches the database comprises files that persistently store the data (col. 9, lines 12-13).

Regarding on claim 7, Hapner teaches the number of tasks accepted from the task sources is arbitrarily large (col. 9, lines 55-60).

Regarding on claim 8, Hapner teaches the number of task sources from which tasks are accepted is arbitrarily large (col. 17, lines 55-60).

Regarding on claim 9, Hapner teaches the regions are organized into contention spaces, the number of contention spaces being no less than the number of available processors (col. 9, lines 55-60).

Regarding on claim 10, Hapner teaches each of the jobs requires write access to data in no more than one of the contention spaces (col. 9, lines 55-60).

Regarding on claim 11, Hapner teaches the number of contention spaces is equal to the number of available processors (col. 9, lines 55-60).

Regarding on claim 12, Hapner teaches the organization of regions into contention spaces maximizes the throughput of the available processors in executing the jobs (col. 9, lines 55-60).

Regarding on claim 13, Hapner teaches the contention spaces are assigned dynamically to processors to maximize the throughput of the available processors (col. 9, lines 55-60).

Regarding on claim 14, Hapner teaches the tasks are accepted asynchronously (col. 9, lines 1-3).

Regarding on claim 15, Hapner teaches the tasks are accepted concurrently (col. 9, lines 55-60).

Regarding on claim 16, Hapner teaches the processors do not use shared memory (col. 18, lines 24-26).

Regarding on claim 17, Hapner teaches defining the jobs (task) for each task comprises defining hierarchy of subtasks in which the lowest level of the hierarchy contains the jobs (col. 18, lines 17-25).

Regarding on claim 18, Hapner teaches at least one of the tasks comprises a single job (col. 9, lines 55-60).

Regarding on claim 19, Hapner teaches a job generating a task to be performed (col. 9, lines 55-60).

Regarding on claim 20, Hapner teaches each of the tasks is completed with a certainty that is at least as high as the certainty that data updated in a requested database transaction is not lost once the transaction is committed (col. 9, lines 5-10).

Regarding on claim 21, Hapner teaches the region (database) comprises a single data item (data record) (col. 9, lines 20-30).

Regarding on claim 22, Hapner teaches the region comprises at least a million data items (data records) (col. 9, lines 20-30).

Regarding on claim 23, Hapner teaches the jobs are executed concurrently without having to wait for release of any write locks on any of the regions (col. 9, lines 55-60).

Regarding on claim 24, Hapner teaches more than one of the contention space is associated with one of the processors (col. 9, lines 55-60).

Regarding on claim 25, Hapner teaches each of the processors comprises a physical processor running at least one process (col. 9, lines 55-60).

Regarding on claim 26, Hapner teaches each of the tasks is generated by a user request (col. 9, lines 55-60).

Regarding on claim 27, Hapner teaches each of the contention spaces is associated with at least two processors one of which execute jobs and the other of which performs administrative functions with respect to the associated contention space (col. 9, lines 55-60).

Regarding on claim 32, Hapner teaches millions of regions belong to a contention space (col. 9, lines 55-60).

Regarding on claim 33, Hapner teaches additional jobs are created in connection with the execution of the jobs (col. 9, lines 55-60).

Regarding on claim 34, Hapner teaches jobs (task) are created by the additional jobs (col. 9, lines 55-60).

Regarding on claim 35, Hapner teaches the creation of the additional jobs is dependent on data read from the database (make file 72) in executing the jobs (col. 9, lines 55-60).

Regarding on claim 36, Hapner teaches additional jobs (sub-tasks) are created in connection with the execution of the jobs (overall task), and a process running on one of the processors execute the jobs and created the additional jobs, and in which at least some of the additional jobs are distributed among contention spaces served by other processor (col. 9, lines 55-60).

Regarding on claim 37, Hapner teaches the tasks relate to a commercial transaction (col. 11, lines 50-58).

Regarding on claim 38, Hapner teaches each of the jobs is assigned in index associated with the corresponding contention space (col. 9, lines 55-60).

Regarding on claim 39, Hapner teaches the indexes are used to load balance the jobs among processors (col. 9, lines 55-60).

Regarding on claim 40, Hapner teaches the database includes database units that are distributed among different physical location (col. 9, lines 27-32).

Regarding on claim 41, Hapner teaches each of the jobs comprises steps sub-tasks (col. 9, lines 27-32).

Regarding on claim 42, Hapner teaches execution of a job includes executing a portion of the steps, committing a database transaction representing those steps, and repeating until the job is completed (col. 9, lines 27-32).

Regarding on claim 43, Hapner teaches upon a failure to complete any portion of the steps, restarting the execution at the first step of the failed portion with at least the same level of certainty that the job will be completed as the certainty that data written in a requested transaction is not lost once the transaction is committed (col. 18, lines 35-41).

Regarding on claim 48, Hapner teaches synchronizing the executions of synchronization group of jobs to ensure correctness of results (col. 9, lines 27-32).

Regarding on claim 49, Hapner teaches the synchronizing includes assigning to each of the jobs of a synchronization group tag that identifies them as members of the group (col. 9, lines 27-32).

Regarding on claim 50, Hapner teaches the synchronizing includes assigning to each of the jobs of a synchronization group a quorum fraction representing the job's proportion of participation in the group (col. 9, lines 27-32).

Regarding on claim 51, Hapner teaches the jobs are not executed until all of the jobs in the synchronization group are ready for execution by a processor (col. 9, lines 27-32).

As to claim 52, Hapner discloses apparatus comprising:

A database that stores data persistently (col. 7, lines 15-16);

A job processing mechanism that (1) accepting an arbitrarily large number of tasks asynchronously from an arbitrarily large number of task sources, at least some of the tasks having competing requirements for use of regions of database (multiples

threads competing for the resources of both database cache 160 and the persistence database portion 164) (col. 9, lines 54-65).

Hapner does not explicitly teaches data included in a given region not being available for simultaneous access for writing by more than one of the tasks having competing requirements and the data and data included in different regions being available for simultaneous access for writing by more than one of the tasks competing requirement, (2) organizes the regions into non-conflicting contention spaces each associated with different available processor, (3) decomposes each of the tasks into jobs each of which require a write access to regions belonging to no more than one of the contention spaces, and (3) distributes the jobs to the corresponding contention spaces for concurrent execution by the associated processor. Cheriton discloses data included in a given region not being available for simultaneous access for writing by more than one of the tasks having competing requirements and the data and data included in different regions being available for simultaneous access for writing by more than one of the tasks competing requirement (...a read operation or an additional shared ownership mode which allows multiple processors and caches to read the data concurrently...) (col. 6, lines 21-59). This suggests some part of the data cannot be access based on an exclusive lock and other data can be accessed concurrently by plurality of processors. Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify teaching Hapner to include some part of the data cannot be access based on a exclusive lock and other data can be accessed concurrently by plurality of processors as disclosed by Cheriton in order to

allow the multiple threads to access data. Hapner does no explicitly disclose (2) organizes the regions into non-conflicting contention spaces each associated with different available processor, (3) decomposes each of the tasks into jobs each of which require a write access to regions belonging to no more than one of the contention spaces, and (3) distributes the jobs to the corresponding contention spaces for concurrent execution by the associated processor. However, Somani discloses (2) organizes the regions into non-conflicting contention spaces each associated with different available processor, (3) decomposes each of the tasks into jobs each of which require a write access to regions belonging to no more than one of the contention spaces, "as a user writes an INSIGHT program 272 defining application tasks and the number of processor used for each task...Each configuration specification file defines a set of generic processors and the jobs partitioned among them" (col. 27, lines 56-62). This teaches a write task is associated with one processor and process lock down a part of the database for writing. Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify Hapner and Cheriton to include the one task for one processor as taught by Somani to allow one processor to read or write the data into a part of the database to keep the data consistency. Furthermore, Fleischman also discloses (4) distributes the jobs to the corresponding contention spaces for concurrent execution by the associated processor. The term "simultaneously" is used in the context known to programmers similar to "multithreading" i.e. multiple threads do not execute in perfectly simultaneous manner unless the server has parallel processor (col. 5, lines 8-20) and multiple small "read" and "write"

commands to a disk drive…(col. 5, lines 33-36). Fleischman disclose the execution of the multiple write threads, and each of the processor is assign to a specify write to a specific portion of the database. Furthermore, in the write thread no other processor to be able to access to the lock region which is accessed by other processor. Therefore, Fleischman implicitly discloses the concept of assigning the assigning each of the region to a available processor, each of the regions being assignable to any of the processors, defining, for each of the tasks, jobs each of which requires write access to regions that are to be accessed by no more than one of the processors and distributing the jobs for concurrent execution by the associated processors. Therefore, it would have been obvious to one ordinary skill in art at the time of the invention was made to modify both Hapner, Cheriton Somani system to include executing of the multiple write threads, and each of the processor is assigned to a specify write to a specific portion of the database and no other processor be able to access to the lock region which is accessed by other processor in the write thread as disclosed by Fleischman to resolve any conflict of accessing the database.

6. Claims 57-61 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hapner et al. (US. Patent No. 5,727,203) in view of Zaiken et al. (US. Patent No. 5,907,848).

Regarding on claim 57, Hapner teaches a physical article or object bearing instruction to cause a processor to execute a job a job (threads), the job requiring access (write to the database 159) (col. 9, lines 25-30) to a region of a database that

stores data persistently (persistence data portion 164) (col. 9, lines 55-60), the job including instruction and pointers to data in the region of the database (col. 9, lines 55-60);

Hapner teaches "as corresponding to multiples threads competing for the resource of both the database cache 160 and the persistence database portion 164" (col. 9, lines 55-60). Hapner does not explicitly teach the job that is executed being selected from a contention space of jobs identified by an index, the jobs in the contention space having competing requirement to write into the region of the database, the index distinguish the contention space from other contention spaces of jobs that do not have competing requirements to write into the region of the database. On the other hand, Zaiken teaches an index that identifies a contention space of jobs that have competing requirement to write into the region of the database, the index distinguish the contention space from other contention spaces of jobs that do not have competing requirements to write into the region of the database "as corresponding to if the file name in the records 20 matches a filename in the selected template 28, the transaction monitor program then searches for any existing index files 30 having job identifier data equal to the job identifier data in the record 20" (col. 11, lines 39-43). This teaches the index identifying the job in the index files to distinguish by comparing the job identifier in the file. Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify Hapner system to include comparing the job identifier in the index files of Zaiken on order to distinguish the jobs by comparing the job identifier in order to process the requested job.

Regarding on claim 58, Hapner teaches the stored data includes items of the database comprises objects in an object database (col. 9, lines 1-3).

Regarding on claim 59, Hapner teaches the stored data includes data items that are provided as objects to an object-oriented application (col. 7, lines 25-30).

Regarding on claim 60, Hapner teaches an objected relational broker provides persistent storage of objects for an object-oriented application (col. 7, lines 34-35).

Regarding on claim 61, Hapner teaches the data is stored in a relational database with object-oriented extensions (col. 7, lines 1-10).

7. Claims 68-83 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hapner et al. (US. Patent No. 5,727,203) in view of Fleischman (US. Patent No. 6,507,847 B1).

Regarding on claim 68, Hapner teaches a method comprising:

Maintaining a database that stores data persistently (col. 7, lines 15-16) and provides a primary level of guarantee that data written in a request transaction is not lost once the transaction is committed (col. 10, lines 11-22),

Accepting tasks from the task source for concurrent execution by multiple processors, at least some of the tasks having conflicting requirements to write into the same region of the database (col. 9, lines 54-65), and

Hapner teaches, "a mutex is created to corresponding to a piece of code, a portion data, some state, etc... when a thread has locked a mutex, it is said to "own" the locked mutex. In order for other threads to own the mutex, the first thread (i.e. the

thread that locked the mutex) must unlock it. Thus mutexes provide a mechanism by which the programmer can control the serialization of multiple threads, ensuring that steps occur in a desired order and that the state corresponding to the mutex is maintained in a consistent manner" (col. 10, lines 11-22). This teaches the mutex lock is the guarantee that the thread holding the job will be executed and no data is lost. Hapner does not explicitly teaches providing a software mechanism that guarantees, as least to the primary level of guarantee, that the tasks will be executed without loss of data and without occurrence of any actual conflict with respect to the region of the database. Further more, Fleischman discloses providing a software mechanism that guarantees, as least to the primary level of guarantee, that the tasks will be executed without loss of data and without occurrence of any actual conflict with respect to the region of the database (a large number of threads can be executing in simultaneously within the database. The term "simultaneously" is used in the context known to programmers similar to "multithreading" i.e. multiple threads do not execute in perfectly simultaneous manner unless the server has parallel processor (col. 5, lines 8-20) and multiple small "read" and "write" commands to a disk drive…(col. 5, lines 33-36). Fleischman disclose the execution of the multiple write threads, and each of the processor is assign to a specify write to a specific portion of the database. Furthermore, in the write thread no other processor to be able to access to the lock region which is accessed by other processor. Therefore, Fleischman implicitly discloses the concept of assigning the assigning each of the region to a available processor, each of the regions being assignable to any of the processors, defining, for

each of the tasks, jobs each of which requires write access to regions that are to be accessed by no more than one of the processors and distributing the jobs for concurrent execution by the associated processors These multiple write thread execution in concurrently will guarantee that tasks will be executed without the loss of data and without occurrence of any actual conflict with respect to the region of the database. Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify Hapner system to include multiple write threads concurrently execution with different portions of data as taught by Fleischman in order guarantee that all tasks will be executed without conflict and loss of data.

Regarding on claim 69, Hapner teaches the stored data includes data items of the database that comprise object in an object database (col. 9, lines 1-3).

Regarding on claim 70, Hapner teaches the stored data includes data items that are provided as objects to an object-oriented application (col. 7, lines 25-30).

Regarding on claim 71, Hapner teaches an object relational broker provides persistent storage of objects-oriented application (col. 7, lines 34-35).

Regarding on claim 72, Hapner teaches the data is stored in a relational database with object-oriented extensions (col. 7, lines 1-10).

Regarding on claim 73, Hapner teaches sending to the task source acknowledgement of acceptance of the task (col. 9, lines 27-32).

Regarding on claim 74, Hapner teach sending to the task source a notification after completion of the accepted task (col. 9, lines 27-32).

Regarding on claim 75, Hapner teach the task is decomposed into jobs that are executed by different ones of the multiple processors in a manner that prevents any actual conflict between jobs a conflict upon complete execution of all tasks (col. 9, lines 27-32).

Regarding on claim 76, Hapner teaches the jobs are subject to a synchronization mechanism that enables a determination of the completion of a task (col. 9, lines 27-32).

Regarding on claim 77, Hapner teaches the synchronization mechanism includes a tag that identifies a job as participating in a group of jobs (col. 9, lines 55-60).

Regarding on claim 78, Hapner teaches the synchronization mechanism includes a quorum fraction that represents the job's proportion of participating in the group (col. 9, lines 55-60).

Regarding on claim 79, Hapner teaches determining whether the quorum fractions of all of the jobs of a group add to a complete quorum (col. 9, lines 55-60).

Regarding on claim 80, Hapner teach the task is notified of completion when all of the jobs have been completed (col. 9, lines 27-32).

Regarding on claim 81, Hapner teaches the task is assigned to a contention space (col. 9, lines 55-60).

Regarding on claim 82, Hapner teaches completion notification jobs are assigned for execution in the same contention space as the task (col. 9, lines 55-60).

Regarding on claim 83, Hapner teaches the database comprises an object-oriented database (col. 9, lines 1-3).

8. Claims 28-31 and 44-47 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hapner et al. (US. Patent No. 5,727,203) in view of Cheriton in view of (US. Patent No. 5,666,514) and further in view of Somani et al. (US. Patent No. 5,524,212) and further in view of Fleischman (US. Patent No. 6,507,847 B) and further in view of Murray (US. Patent No. 5,404,521).

Regarding on claim 28, Hapner, Cheriton, Somani and Fleischman do not explicitly teach the distributing of the jobs includes maintaining a queuing system that has a capacity to receive jobs for execution at an arbitrarily large rate by adding processors proportionately to the number of jobs expected to require execution. However, Murray teaches the distributing of the jobs includes maintaining a queuing system that has a capacity to receive jobs for execution at an arbitrarily large rate by adding processors proportionately to the number of jobs expected to require execution "as corresponding to multiprocessor computer system having a plurality of processors (main processor and parallel processors) wherein an application program is separated into different tasks which are executed in parallel by the system processor" (col. 3, lines 36-40). In addition, Murray also teaches, "if there are more than one tasks, the parallel processor will perform the execution for that task" (col. 3, lines 45-54). This teaches other preserved processor to execute the tasks when there are more than one task require for execution. Therefore, it would have been obvious to one ordinary skill in the

art at the time of the invention was made to modify Somani, Cheriton, Hapner and Fleischman to include the queue system to call in for other preserved processors to execute task in the event there are more tasks waiting to execute as taught by Murray in order to enhance system performance in the multiprocessors system.

Regarding on claim 29, Hapner, Cheriton, Somani and Fleischman do not explicitly teach the queuing system includes conceptual rows each of which can receive the jobs; however, Murray teaches the queuing system includes conceptual rows each of which can receive the jobs (col. 5, lines 10-15). Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify both Somani, Cheriton, Hapner and Fleischman to include the queue system to include table having row and column to receives jobs as taught by Murray in order to enhance system performance in the multiprocessors system.

Regarding on claim 30, Hapner, Cheriton, Somani and Fleischman do not explicitly teach each of the row is locked when jobs are being received in the row; however, Murray teaches each of the row is locked when jobs are being received in the row (col. 5, lines 23-26). Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify Somani, Cheriton, Hapner and Fleischman to include row lock in the queue system as taught by Murray in order protect the row from being accessed to enhance system performance in the multiprocessors system.

Regarding on claim 31, Hapner teaches a job can be accepted by the corresponding processor for execution from any of the rows that is not locked (col. 10, lines 11-22).

Regarding on claim 44, Hapner teaches a row is locked only for reading when a processor is accepting jobs from that row (col. 10, lines 11-22).

Regarding on claim 45, Hapner, Cheriton, Somani and Fleischman do not explicitly teach (1) the distributing of the jobs includes maintaining a queuing system that has a capacity to receive jobs for execution at any arbitrarily large rate, (2) the queuing system includes conceptual rows each of which can receive the jobs, and (3) the queuing system includes conceptual columns associated with respective contention spaces, the queuing system comprising a conceptual matrix of cells at the intersections of the rows and columns, in which each of the cells may be read from or written to without conflicting with reads and writes to other cells; however, Murray teaches (1) the distributing of the jobs includes maintaining a queuing system that has a capacity to receive jobs for execution at any arbitrarily large rate, (2) the queuing system includes conceptual rows each of which can receive the jobs, and (3) the queuing system includes conceptual columns associated with respective contention spaces, the queuing system comprising a conceptual matrix of cells at the intersections of the rows and columns, in which each of the cells may be read from or written to without conflicting with reads and writes to other cells (col. 6, lines 30-59). Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify Hapner, Cheriton, Somani and Fleischman to include the queue system to call in for

other preserved processors to execute task in the event there are more tasks waiting to execute as taught by Murray in order to enhance system performance in the multiprocessors system.

Regarding on claim 46, Hapner, Cheriton, Somani and Fleischman do no explicitly teach the rows are associated with source of jobs and the number of rows is sufficient to permit all of the sources of jobs to load jobs into the queue concurrently; however, Murray teaches the rows are associated with source of jobs and the number of rows is sufficient to permit all of the sources of jobs to load jobs into the queue concurrently (col. 6, lines 15-21). Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify Somani, Cheriton and Hapner and Fleischman to include the queue system to load job currently when rows are available for process as taught by Murray in order to enhance system performance in the multiprocessors system.

Regarding on claim 47, Hapner, Cheriton, Somani and Fleischman do no explicitly teach the number of rows is sufficient to permit jobs to be fetched for execution from all of the columns concurrently; however, Murray teaches the number of rows is sufficient to permit jobs to be fetched for execution from all of the columns concurrently (col. 4, lines 59-61). Therefore, it would have been obvious to one ordinary skill in the art at the time of the invention was made to modify Hapner, Cheriton, Somani and Fleischman to include the queue system to load job currently when rows are available for process as taught by Murray in order to enhance system performance in the multiprocessors system.

Conclusion

9. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Contact Information

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Baoquoc N. To whose telephone number is at 571-272-4041, or unofficial fax number for the purpose of discussion (571) 273-4041 or via e-mail BaoquocN.To@uspto.gov. The examiner can normally be reached on Monday-Friday: 8:00 AM – 4:30 PM, EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Breene can be reached at 571-272-4107.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks
Washington, D.C. 20231.

The fax numbers for the organization where this application or proceeding is assigned are as follow:

(571) 273-8300 [Official Communication]

/Baoquoc N To/
Primary Examiner, Art Unit 2162
August 16th, 2008